6r
.235
p. 2

Math

Report No. 235

A DETERMINISTIC PROCEDURE FOR THE DESIGN OF CARRY-SAVE ADDERS AND BORROW-SAVE SUBTRACTERS

by

James E. Robertson

July 5, 1967

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 235

A DETERMINISTIC PROCEDURE FOR THE DESIGN OF CARRY-SAVE ADDERS AND BORROW-SAVE SUBTRACTERS*

by

James E. Robertson

July 5, 1967

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* To be presented at the First Annual IEEE Computer Conference, Chicago, Illinois, September 6-8, 1967.

TABLE OF CONTENTS

A DETERMINISTIC PROCEDURE FOR THE DESIGN OF CARRY-SAVE ADDERS AND BORROW-SAVE SUBTRACTERS

by

James E. Robertson
Department of Computer Science
University of Illinois
Urbana, Illinois

ABSTRACT

A recently developed deterministic procedure will be described, and employed to determine binary carry-save adder and borrow-save subtracter designs. Designs believed to be new as well as those previously known will be developed. A particular structure, expressed in terms of normalized digit sets, leads to three practical algebraic structures. Each of these in turn leads to nine (not necessarily unique) logical designs. Examples will be given and possible application to variable field length systems will be discussed.

1. REVIEW

Historically, the design of carry save adders and borrow save subtracters has been conducted by modification of conventional adders with propagating carry and of subtracters with propagating borrow. As an example, the Boolean equations for the ith digital position of a binary subtracter with propagating borrow are

$$\begin{aligned} s_i &= a_i \oplus m_i \oplus b_i \\ b_{i-1} &= \bar{a}_i b_i \vee b_i m_i \vee m_i \bar{a}_i \ , \end{aligned} \tag{1}$$

in which a_i is a minuend digit, m_i is a subtrahend digit, s_i is a difference digit, and b_i and b_{i-1} are respectively the incoming and outgoing propagating borrow. Equations (1) are the Boolean equivalent of the algebraic expression:

$$-2b_{i-1} + s_i = a_i - m_i - b_i \tag{2}$$

The direct modification to a borrow save subtracter is one of associating with the a_i stored borrow digits α_i, and associating with the s_i stored borrow digits σ_i. Equations (1) then become

$$\begin{aligned} s_i &= a_i \oplus m_i \oplus \alpha_i \\ \sigma_{i-1} &= \bar{a}_i \alpha_i \vee \alpha_i m_i \vee m_i \bar{a}_i \end{aligned} \tag{3}$$

in which, as before, m_i is a subtrahend digit, but now the minuend is represented by the appropriately weighted sum of the digits a_i and α_i, and the difference is represented by the weighted sum of the digits s_i and σ_i. If the digits s_i and σ_i are associated to form a redundantly represented digit s_i^* in accordance with the formula $s_i^* = s_i - \sigma_i$, the possible values of s_i^* are -1, 0, and 1.

A less obvious modification to a borrow save subtracter is achieved by rewriting Equations (1) and substituting as follows:

$$\left.\begin{aligned} s_i &= a_i \oplus m_i \oplus b_i \\ b_{i-1} &= (\overline{a_i \oplus m_i})b_i \vee \bar{a}_i m_i \end{aligned}\right. \tag{4}$$

Let

$$k_i = (\overline{a_i \oplus m_i})b_i$$

$$b_{i-1} = k_i \vee \bar{a}_i m_i$$

$$\left.\begin{aligned} s_i &= (a_i \oplus m_i) \oplus (k_{i+1} \vee \bar{a}_{i+1} m_{i+1}) \\ k_i &= (\overline{a_i \oplus m_i})\circ(k_{i+1} \vee \bar{a}_{i+1} m_{i+1}) \end{aligned}\right\} \tag{5}$$

If, in Equations (5), the input propagating borrow k_{i+1} is replaced by a stored borrow α_{i+1} in association with a_{i+1}, and the output propagating borrow k_i is replaced with a stored borrow σ_i in association with s_i, the Boolean equations for the stored borrow subtracter become

$$\left.\begin{aligned} s_i &= (a_i \oplus m_i) \oplus (\alpha_{i+1} \vee \bar{a}_{i+1} m_{i+1}) \\ \sigma_i &= (\overline{a_i \oplus m_i})\circ(\alpha_{i+1} \vee \bar{a}_{i+1} m_{i+1}) \end{aligned}\right. \tag{6}$$

In Equations (6), the combination $s_i = 0$, $\sigma_i = 1$, cannot occur, hence a redundantly represented digit $s_i^* = -2\sigma_i + s_i$ may assume only the three values -1, 0, and 1.

As is evident from the expression for b_{i-1} in Equations (4), the propagating borrow must necessarily pass through an AND circuit and an OR circuit in each digital position. The stored borrow of the subtracter of Equations (3) is the output of the OR circuit; the stored borrow of Equations (6) is the output of the AND circuit; thus the subtracter structures require equivalent hardware. The relationship $\sigma_i \bar{s}_i = 0$ implied by Equations (6) simplifies the circuitry for conversion of the difference, represented by the s_i and σ_i, to conventional (non-redundant) form, and is therefore preferable if separate circuitry is provided for conversion.

2. THE DETERMINISTIC PROCEDURE

The purpose of the following sections is to describe a deterministic procedure for the design of carry-save adders and borrow-save subtracters. In the case of binary subtracters, which are chosen for specific examples, the procedure is shown to result in not only the two designs described in the previous section, but in new designs as well. The procedure involves three stages of design, as follows:

1) Specification in terms of normalized digit sets,
2) Choice of specific algebraic values, and
3) Logical design with specific binary formats.

For the purposes of this paper, an n-valued digit set is defined as a sequence of n successive integers m, m+1, ..., m+n-1, with m an integer (positive, negative, or zero) and n an integer greater than 1. The digit set is normalized if m=0. The digit set is symmetric if n is odd and $m = -\frac{n-1}{2}$. It follows that for any element x of a symmetric digit set, -x is also an element of the digit set. For a number in conventional form, the digit set associated with each digital position is normalized with n=r, the radix. If $n > r$, the digit set is redundant.

A carry-save adder (or borrow-save subtracter) necessarily requires that the digit set of the sum (or difference) be redundant; otherwise, the representation of the sum (or difference) is unique, which implies that each digit is in general a function of all digits of lesser significance. Only digit sets of minimum redundancy (n=r+1) for the binary case (r=2) are discussed here. Under these restrictions the first stage of design, specified in terms of normalized digit sets, leads to only one possible specification.

In the second stage of design, the choice of the symmetric digit set for the sum (or difference) is considered as well, and leads to two additional algebraic specifications, one being an adder and the other a subtracter. Thus the second stage of design, requiring a choice of specific algebraic values, leads to three algebraic structures, as follows:

1) The normalized carry-save adder,
2) The symmetric carry-save adder, and
3) The symmetric borrow-save subtracter.

For the structures considered here, a first input is in conventional form (n=r=2), the second input is in minimally redundant form (n=r+1=3), and the output (sum or difference) is also in minimally redundant form. Before the third stage (logical design) can proceed, it is necessary to specify precisely how the minimally redundant digits are represented in terms of two binary digits; that is, a binary format is chosen for the redundant digit sets. It is also required that the same format be chosen for the redundant output and the redundant input, since commonly the output of one calculation becomes the input for the next, as in, for example, recursive steps of a multiplication.

Under these conditions, seventy-two formats, or logical designs, are possible for each of the three algebraic structures. Formats of the same PN type, however, are considered equivalent, since permutation is equivalent to interchanging names of the two binary digits employed, and negation of a format digit is equivalent to replacing it by its complement wherever it appears in the design equations. There are then only nine PN types of logical designs for each of the three structures. For brevity, the nine PN types of designs will be described here only for the third algebraic structure, the symmetric borrow-save subtracter.

3. CHOICE OF NORMALIZED DIGIT SETS

The usual procedure for normalized digit set design is one of specifying the output and input digit sets, and then determining a structure meeting these specifications.

Figure 1 shows a structure consisting of two addition tables, or levels, (each shown by a box) which meets the specifications of the minimally redundant carry-save adder considered here. It is possible to show, for the given specifications, that two levels are necessary and sufficient, and that the digit sets must be chosen with the number of values of Figure 1. The following trivial observations are necessary:

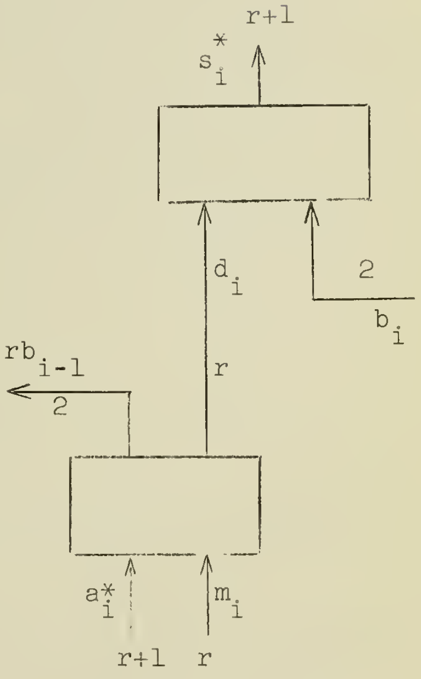

Figure 1.

1) For any addition table, the number of values of the sum digit set is one less than the sum of the numbers of values of the addend and augend digit sets.
2) The transfer input set labelled b_i has the two normalized values 0, 1; the transfer output set rb_{i-1} has the values obtained by multiplying those of b_i by the radix r; namely 0, r.
3) The composite digit set representative of the output (sum) of the lower addition table is $rb_{i-1} + d_i$. If this digit set is to consist of successive integers, the digit set for d_i must have at least r values.
4) The composite digit set $rb_{i-1} + d_i$ will have a number of values which is at most the product of the numbers of values of rb_{i-1} and of d_i. The maximum number of values is achieved only if the number of values of d_i is precisely r; if the digit set of d_i is redundant, the composite set will have fewer values than the product of those of rb_{i-1} and d_i.

The first observation implies that the sum of the numbers of values of the digit sets of d_i and b_i is $r + 2$. Since no digit set has less than two values, b_i must have at least two values. The third observation indicates that d_i must have at least r values, hence the choices shown for d_i and b_i are the only ones possible.

Since the values of d_i and b_i are not compatible with the specifications for the addend and augend of the carry-save adder, another addition table is necessary. From observation 4, the number of values of the output digit set is 2r, hence the two inputs will have a total number of values equal to $2r + 1$. This then permits one input of r values, and a second input of $r + 1$ values, consistent with the desired specifications.

4. EXTENSION TO SYMMETRIC DIGIT SETS

Since each normalized digit set has zero as its least value, all digit sets appearing on Figure 1 are normalized, and hence completely specified as indicated in line 1 of Table 1. If the digit sets of s_i^* and a_i^* are symmetric, however, two solutions are possible, as shown in lines 2 and 3 of Table 1 ($\bar{1}$ represents -1).

	s_i^*	b_i	d_i	rb_{i-1}	m_i	a_i^*
normalized adder	0,1,2	0,1	0,1	0,2	0,1	0,1,2
symmetric adder	$\bar{1}$,0,1	0,1	$\bar{1}$,0	0,2	0,1	$\bar{1}$,0,1
symmetric subtracter	$\bar{1}$,0,1	$\bar{1}$,0	0,1	$\bar{2}$,0	$\bar{1}$,0	$\bar{1}$,0,1

Table 1. Values of Digit Sets for Three Algebraic Structures.

In Table 1, values for the digit sets must be chosen so that the equations $s_i^* = b_i + d_i$ and $rb_{i-1} + d_i = m_i + a_i^*$ are always satisfied. A structure is called an adder if the transfer digit b_i and the addend digit m_i are both positive; if both are negative the structure is called a subtracter, with m_i a digit of the subtrahend.

5. CHOICE OF BINARY FORMATS FOR s_i^* AND a_i^*

The first step in logical design is the choice of binary state assignments to be considered. This is a tedious but straightforward process, which may proceed as follows:

1) List the 24 ways of assigning the four states of two binary digits to four digital values, with the fourth value as yet unspecified.
2) For each of three groups, list the eight state assignments equivalent under permutation and negation. Choose one assignment from each group.
3) For each of the three state assignments resulting from step 2, list four sets of digital values by assigning to the previously unspecified fourth digital value the values 1, 0, $\bar{1}$, and "don't care".
4) Among the twelve assignments resulting from step 3, identify the three pairs equivalent under permutation and negation, and delete one of each pair. The equivalence results from the fact that the same digital value is assigned to two states during step 3. These two states can then be interchanged and the assignment will then be equivalent to another of the twelve assignments of step 3.

The procedure outlined above results in nine assignments, none of which are equivalent under permutation and negation. One such set of nine assignments for the digital values $\bar{1}$, 0, and 1 is given in Table 2. The assignments are numbered in Table 2 for future reference.

State	Digital Values								
0 0	0	0	0	0	D.C.	0	0	1	$\bar{1}$
0 1	1	1	1	1	1	1	1	1	1
1 0	$\bar{1}$	D.C.	0	$\bar{1}$	0	1	$\bar{1}$	0	0
1 1	0	$\bar{1}$	$\bar{1}$	D.C.	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
Design Number	1	2	3	4	5	6	7	8	9

Table 2. Assignments of Digital Values to Four States.

For the logical designs to follow, the two binary digits which represent a_i^* will be called α_i and a_i; those which represent s_i^* will be called σ_i and s_i. For reasons given before, a_i^* and s_i^* are always consistently represented.

6. DETAILED LOGICAL DESIGNS

Simplified Boolean equations for the detailed logical designs for each of the nine assignments are given below.

Design Number	Lower Addition Table	Upper Addition Table
1	$b_{i-1} = \bar{a}_i m_i \vee \bar{a}_i \alpha_i \vee \alpha_i m_i$	$\left.\begin{array}{l}\sigma_i = b_i \\ s_i = d_i\end{array}\right\}$ or $\left\{\begin{array}{l}\sigma_i = \bar{d}_i \\ s_i = \bar{b}_i\end{array}\right.$
	$d_i = \alpha_i \oplus a_i \oplus m_i$	
2	$b_{i-1} = \alpha_i \vee \bar{a}_i m_i$	$\sigma_i = b_i \bar{d}_i$
	$d_i = a_i \oplus m_i$	$s_i = b_i \oplus d_i$
3	$b_{i-1} = \alpha_i a_i \vee \bar{a}_i m_i$	$\sigma_i = b_i$ or $\sigma_i = \bar{d}_i$
	$d_i = a_i \oplus m_i$	$s_i = b_i \oplus d_i$
4	$b_{i-1} = \alpha_i \vee \bar{a}_i m_i$	$\sigma_i = b_i \bar{d}_i$
	$d_i = m_i \oplus (\alpha_i \vee a_i)$	$s_i = \bar{b}_i d_i$
5	$b_{i-1} = \alpha_i (a_i \vee m_i)$	$\sigma_i = b_i \vee \bar{d}_i$
	$d_i = a_i \oplus m_i$	$s_i = b_i \oplus d_i$
6	$b_{i-1} = \alpha_i a_i \vee \bar{\alpha}_i \bar{a}_i m_i$	$\left.\begin{array}{l}\sigma_i = b_i \bar{d}_i \\ s_i = b_i \oplus d_i\end{array}\right\}$ or $\left\{\begin{array}{l}\sigma_i = b_i \oplus d_i \\ s_i = b_i \bar{d}_i\end{array}\right.$
	$d_i = m_i \oplus (\alpha_i \vee a_i)$	
7	$b_{i-1} = \alpha_i \vee \bar{a}_i m_i$	$\sigma_i = b_i \bar{d}_i$
	$d_i = m_i \oplus (\alpha_i \vee a_i)$	$s_i = \bar{b}_i d_i$
8	$b_{i-1} = \alpha_i (a_i \vee m_i)$	$\sigma_i = b_i \vee \bar{d}_i$
	$d_i = m_i \oplus (\bar{\alpha}_i \vee a_i)$	$s_i = b_i \bar{d}_i$
9	$b_{i-1} = \alpha_i a_i \vee a_i m_i \vee \bar{\alpha}_i \bar{a}_i$	$\left.\begin{array}{l}\sigma_i = b_i \vee \bar{d}_i \\ s_i = b_i \oplus d_i\end{array}\right\}$ or $\left\{\begin{array}{l}\sigma_i = \overline{b_i \oplus d_i} \\ s_i = \bar{b}_i d_i\end{array}\right.$
	$d_i = m_i \oplus (\bar{\alpha}_i \vee a_i)$	

Table 3. Boolean Equations for Symmetric Borrow-Save Subtracters.

The combined equations for design 1 can be written as

$$s_i = \alpha_i \oplus a_i \oplus m_i$$

$$\sigma_{i-1} = \bar{a}_i m_i \vee \bar{a}_i \alpha_i \vee \alpha_i m_i \ .$$

Design 1 is therefore identical with the long-known stored borrow subtracter of Equations (3). The state assignment for design 1 is equivalent to giving α_i a weight of -1 and a_i a weight of +1 (Table 2). Correspondingly σ_{i-1} has weight -2 and s_i has weight +1, so that

$$-2\sigma_{i-1} + s_i = a_i - m_i - \alpha_i \ ,$$

which is equivalent to Equation (2) interpreted for the stored-borrow modification.

The combined equations for design 2 can be written as

$$s_i = a_i \oplus m_i \oplus (\alpha_{i+1} \vee \bar{a}_{i+1} m_{i+1})$$

$$\sigma_i = (\overline{a_i \oplus m_i})(\alpha_{i+1} \vee \bar{a}_{i+1} m_{i+1})$$

indicating that design 2 is identical with the long-known stored-borrow subtracter of Equations (6). The state assignment for design 2 is equivalent to giving α_i and σ_i weights of -2, and a_i and s_i weights of +1, with the value -2 excluded (corresponding to $\sigma_i = 1$, $s_i = 0$).

Designs 4 and 7, which are identical, are minor variations of design 2. In these designs, $b_i \oplus d_i = \sigma_i \vee s_i$, whereas in design 2, $b_i \oplus d_i = s_i$. In effect the OR operation needed to complete the EXCLUSIVE OR in the upper addition table is supplied in the formation of d_i in the lower addition table. Design 8, if a_i and s_i are complemented, becomes the Boolean dual of designs 4 and 7. Design 5 is equivalent, under negation of a_i, to the Boolean dual of design 2.

Designs 6 and 9 appear too complicated to warrant further discussion.

Design 3 is believed to be new. The state assignment for design 3 is equivalent to α_i and a_i being, respectively, the sign and magnitude of a_i^*. This design is therefore referred to as the stored sign subtracter.

7. FURTHER DESIGN CONSIDERATIONS

Frequently when stored-borrow subtracters are employed, separate circuitry is provided for conversion, or assimilation, of the redundantly represented digits to conventional form. In Figure 1 for a binary subtracter, the lower addition table can be used for assimilation with a_i^* the digit to be assimilated, m_i the incoming propagating borrow, rb_{i-1} the outgoing propagating borrow, and d_i the assimilated digit. Thus, the complexity of the assimilation circuitry is indicated by the design equations for the lower addition table in Table 3. The relative difficulty of assimilation for design 1, discussed earlier, is clearly apparent. Designs 2 and 5 are the simplest, with designs 3, 4, 7, and 8 only slightly more complex.

Although conversion of a subtracter into an adder is usually performed by complementation of each digit m_i, with borrow insertion in the least significant digit, an alternative approach is advantageous in some circumstances. Let s, a, and m be the numbers represented by all digits s_i^*, a_i^*, and m_i, respectively. If the values of s_i^* and a_i^* are symmetric, then -s and -a can be formed by replacing s_i^* by $-s_i^*$ and a_i^* by $-a_i^*$ in each digital position. Since $s = a - m$, the result of these negations is $-s = -(-a - m) = a + m$, that is, an addition is performed. This approach has potential advantages in some variable field length operations, since it does not require any special operations on the least significant digit. The stored sign subtracter of design 3 is simplest for this operation, since negation of each digit requires only the complementation of its sign digit.

An advantage of the deterministic procedure described is that statistical calculations can be made at the first stage of design, and, if interpreted properly, apply to all descendents of that design. For example, the probability of the center digit (1 for the normalized digit set; 0 for the symmetric digit set) of the sum digit set s_i^* is always $\frac{1}{2}$ if the two values of m_i are equally likely, independent of the probabilities of the values of a_i^*.

8. SUMMARY AND CONCLUSIONS

A deterministic procedure has been described which begins with a single normalized digit set structure for a carry-save adder which leads to three algebraic structures. Each of these three result in nine logical designs, one group of which has been described. Of the nine, two have been known for many years, four are minor variations of these, two are new but appear impractical, and one is new and appears to be of potential value for some variable field length systems.

The digit set approach has been employed[1] for the study of more complicated binary structures, including three level adders in which both addend and augend are three-valued, and for the study of higher radix structures. The carry-save adder described here is simple and unusually tractable in comparison.

BIBLIOGRAPHY

1. Rohatsch, Fredrich A., "A Study of Transformations Applicable to the Development of Limited Carry-Borrow Propagation Adders", Ph.D. Thesis, University of Illinois, Urbana, Illinois, June 1967.

2. Atkins, Daniel E., "The Theory and Implementation of SRT Division," M.S. Thesis, University of Illinois, Urbana, Illinois, June, 1967, pp. 49-53.

ACKNOWLEDGMENTS

The use of transformations from normalized digit sets and the development of systematic procedures for finding limited carry propagation adders subject to specified operand and sum digit sets is based on the work of Dr. Fred Rohatsch. The author is grateful to Dr. Rohatsch for many productive discussions on the subject matter of this paper.

The inspiration for the research on which this paper was based was provided by the discovery in July, 1965, of a stored sign symmetric adder (Table 1, line 2 with the format of design 3, Table 2) while the author was a summer employee of Bell Telephone Laboratories, Whippany, New Jersey. The author is grateful to his associates during that period for many stimulating conversations relevant to this paper.

The stored sign subtracter (Design 3 of Table 3) is being used in the arithmetic unit of Illiac III at the University of Illinois. The author is grateful that this design is encountering the practical tests of a physical realization in hardware. The onerous tasks of conversion from a logical design into a physical realization have been performed by Mr. Daniel E. Atkins.[2]